A Framework for Learning Phonological Maps as Logical Transductions

Tatevik Yolyan
Department of Linguistics / Rutgers University / New Brunswick, NJ
tatevik.yolyan@rutgers.edu

Abstract

This paper presents an approach to learning a subclass of phonological maps that can be expressed with non-recursive quantifier-free logical transductions within the framework of Boolean Monadic Recursive Schemes (BMRS). Building on previous work on phonological learning with partially-ordered hypothesis spaces, this paper shows that similar structures can be used in learning phonological maps when these maps are represented by logical transducers over unconventional string models. This paper demonstrates how a model-theoretic framework can be used for computational learning of *phonological generalizations*.

1 Introduction

Input Strictly Local (ISL) functions characterize phonological maps in which sounds undergo change based on local information in the input (Chandlee, 2014; Chandlee and Heinz, 2018). In other words, every ISL function has a parameter k such that for every x in the input string, the information needed to determine the output of x is within some window of size k around x. Consider the postnasal voicing map below from Zoque, in which voiceless stops become voiced when they are immediately preceded by a nasal (Wonderly, 1951). The input-output pair of words in Figure 1 illustrates why this map is ISL-2.

pama	‘clothing’	m**b**ama	‘my clothing’
tatah	‘father’	n**d**atah	‘my father’
kama	‘cornfield’	ŋ**g**ama	‘my (corn)field’
hayah	‘husband’	nhayah	‘my husband’

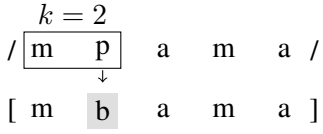


Figure 1: Postnasal voicing in Zoque is ISL-2.

The ISL class of functions is learnable from positive data; for every ISL function f, it is possible to learn a finite state transducer which computes f from a characteristic sample $S \subseteq f$ of input-output pairs of strings (Chandlee, 2014; Chandlee et al., 2014). The purpose of this paper is to revisit learning from a model-theoretic perspective, where the goal is to learn ISL functions as *logical* transductions from pairs of input-output string *models*. While finite state transducers traditionally operate over string representations, logical transducers operate over more enriched representations. This paper uses ‘unconventional’ string models (Strother-Garcia et al., 2016; Chandlee et al., 2019) which represent strings with phonological features.

One of the main ideas presented in this paper is that we can learn logical transductions from input-output pairs of string models by using partially-ordered hypothesis spaces. Similar spaces have been used for various learning problems within phonology (Rawski, 2021; Chandlee et al., 2019; Tesar, 2013; Heinz et al., 2012; Heinz, 2010). A common theme among them is that a partially-ordered hypothesis space encodes entailment relations that are relevant for learning. These entailments allow a large space to be pruned efficiently with a small number of data points. This paper shows that a similar approach can be applied to the problem of learning logical transducers. This paper culminates with a demonstration of how a logical transducer for postnasal voicing can be learned using a very small number of input-output pairs.

This work closely relates to previous work on model-theoretic learning of phonotactic constraints from Chandlee et al. (2019) and Rawski (2021). More specifically, this paper shows that the problem of learning the environment that triggers a particular feature to undergo change employs the same structure as learning banned k-factors in a grammar. In the case of postnasal voicing, for example, Zoque has a phonotactic constraint *[+nas][-voi,-

cont] which says a voiceless stop cannot be immediately preceded by a nasal. This constraint also expresses the *environment* where voicing takes place: if an underlying form violates *[+nas][-voi,-cont], the second sound undergoes voicing in order to repair the violation. The partially-ordered hypothesis space used to learn the surface constraint *[+nas][-voi,-cont] can therefore be adapted to learning the environment where a voiceless sound becomes voiced. As such, the procedure presented here is a means of *lifting* phonotactic learning to learning phonological maps via logical transductions.

A broader contribution of this paper is a demonstration of how this model-theoretic approach can be used to learn phonological generalizations from a sample of underlying and surface form pairs. To that end, the style of this paper is mainly expository, and the learning problem is abstract rather than formally-defined. This paper moreover focuses on length-preserving and order-preserving ISL functions because they are a natural starting point for motivating the larger research program of model-theoretic phonology. The method introduced here can be extended to more complex function classes, as well as more complex representations.

The structure of this paper is as follows. Section 2 provides the relevant background and definitions of string models, logical transductions, subfactors, and partially-ordered sets (posets). Section 3 discusses previous work on phonological learning with posets, with emphasis on the Bottom-Up Feature Inference Algorithm (BUFIA; Chandlee et al., 2019). Section 4 presents the learning goal and procedure, and Section 4.4 illustrates these ideas using postnasal voicing as a case study. Future extensions of this work are discussion in Section 5.

2 Preliminaries

Given an alphabet Σ, a string/word is a finite concatenation of symbols in Σ. The set of all such strings, along with the empty string λ, is denoted Σ^*. For each $w \in \Sigma^*$, we let w_i denote the i^{th} character in w, and $|w|$ denote the length of w. This section presents definitions of relational structures and string models which are used to represent words in Σ^*, and ultimately functions over Σ^*. These structures are then enhanced with phonological features over which phonological maps are defined.

2.1 String Models

Definition 1. *A **signature** is a collection of relation symbols $\{R_i\}_{i\leq m}$, and function symbols $\{f_i\}_{i\leq n}$.*[1]

Definition 2. *Given a signature $\mathcal{S}$, an $\mathcal{S}$-**structure***

$$\mathcal{M} = \langle D, \{R_i^{\mathcal{M}}\}_{i\leq m}, \{f_i^{\mathcal{M}}\}_{i\leq n}\rangle$$

consists of a domain D, a relation $R_i^{\mathcal{M}} \subseteq D^k$ for each k-ary relation symbol R_i, and a function $f^{\mathcal{M}} : D^k \to D$ for every k-ary function symbol f. $Struc(\mathcal{S})$ denotes the set of $\mathcal{S}$-structures.

Structures are built over a signature by specifying a domain and interpretations for the relation and functions symbols. Every word in Σ^* is associated with a particular kind of structure called a *string model* built over a signature consisting of unary predicates $\{\rtimes, \ltimes, P_\sigma\}_{\sigma\in\Sigma}$ and successor and predecessor functions. An alphabet and a signature over that alphabet are often designated by the same symbol; the string models built over an alphabet Σ are called Σ-structures.

This paper looks at a specific type of string model which we refer to here as *monadic* string models, following Chandlee and Lindell (forth.).

Definition 3. *For an alphabet Σ and word $w \in \Sigma^*$, a **monadic string model** for w is a Σ-structure*

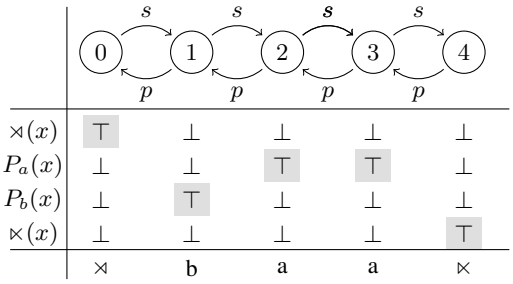
$$\mathcal{M}(w) := \langle D, \rtimes, \ltimes, P_\sigma, s, p\rangle_{\sigma\in\Sigma}$$

where $D = \{0, \ldots, |w| + 1\}$ is a set of indices; each $P_\sigma : D \to \{\top, \bot\}$ is a monadic predicate s.t. $P_\sigma(i) = \top$ iff $w_i = \sigma$; $\rtimes(i) = \top$ iff $i = 0$; $\ltimes(i) = \top$ iff $i = |w| + 1$; $s, p : D \to D$ are successor and predecessor functions over D; for each $i \in \{1, \ldots, |w|\}$, exactly one σ is s.t. $P_\sigma(i) = \top$.

The predicates P_σ indicate which symbol of the alphabet appears at an index, the symbols $\rtimes$ and $\ltimes$ indicate the left and right edges of the word, and p and s capture the linear ordering of the symbols. The monadic model for the string 'baa' over the alphabet $\{a, b\}$ is illustrated in Figure 2.

String models can be further enriched with other relations/functions to represent autosegmental structure (Lambert and Rogers, 2020; Chandlee and Jardine, 2019a; Jardine, 2017), syllable structure (Strother-Garcia, 2019, 2018; Strother-Garcia et al., 2016), prosodic structure (Dolatian, 2020), and syntactic structure (Rogers, 2003). String

[1] The concept of 'signature' is also commonly referred to in model theory literature as 'vocabulary' (e.g. Libkin, 2004) or 'language' (e.g. Marker, 2002).



$\rtimes(x)$	⊤	⊥	⊥	⊥	⊥
$P_a(x)$	⊥	⊥	⊤	⊤	⊥
$P_b(x)$	⊥	⊤	⊥	⊥	⊥
$\ltimes(x)$	⊥	⊥	⊥	⊥	⊤
	⋊	b	a	a	⋉

Figure 2: $\mathcal{M}(baa)$: monadic model of the string 'baa'.

models have been used to characterize subregular classes of phonological grammars (Lambert and Rogers, 2020; Rogers and Lambert, 2019; Rogers and Pullum, 2011) and to develop inference algorithms for these grammars (Rawski, 2021; Chandlee et al., 2019; Strother-Garcia et al., 2016; Vu et al., 2018).

2.2 Logical Transductions

The focus of this paper is on phonological *maps*. We represent the underlying and surface forms of words with string models, and represent phonological maps as *relations* between string models. Consider a string function $f : \Sigma^* \to \Gamma^*$. For every input string $w \in \Sigma^*$ and corresponding Σ-structure $\mathcal{M}(w)$, there is an output string $f(w)$ and corresponding Γ-structure $\mathcal{M}(f(w))$. We call $\Sigma = \langle P_\sigma, \rtimes, \ltimes, p, s\rangle_{\sigma\in\Sigma}$ the *input signature* and $\Gamma = \langle P_\gamma, \rtimes, \ltimes, p, s\rangle_{\gamma\in\Gamma}$ the *output signature*, and capture the relationship between input and output string models with a map that transforms Σ-structures into Γ-structures by means of logical formulas. In particular, the predicates $\{P_\gamma\}_{\gamma\in\Gamma}$ in the output Γ-structure are associated with logical formulas over the predicates $\{P_\sigma\}_{\sigma\in\Sigma}$ in the input Σ-structure. These maps are called *logical transductions* (Courcelle, 1994; Engelfriet and Hoogeboom, 2001; Courcelle and Engelfriet, 2012). Logical transductions over monadic string models are quantifier-free (Chandlee and Jardine, 2019b, 2021; Chandlee and Lindell, forth.).

As an example, consider the simple function f expressed by the rewrite rule $a \to b/b\_\_$, applied over the alphabet $\Sigma = \{a, b\}$. The logical transducer which expresses f transforms a Σ-structure $\langle D, \rtimes, \ltimes, P_a, P_b, p, s\rangle$ into a Σ-structure $\langle D, \rtimes, \ltimes, P'_a, P'_b, p, s\rangle$, where P'_a and P'_b are expressed as logical formulas over the *signature of the input model*, as in (1). The predicates P_a and P_b express when an index in the input string model carries an 'a' or 'b' symbol, respectively. The predicates P'_a and P'_b express when an index in the *output* string model carries an 'a' or 'b'. Since f is a length-preserving function, the input and output models have the same domain, linear ordering, and boundary predicates $\rtimes$ and $\ltimes$.

input	⋊	b	a	a	⋉
	0	1	2	3	4
$\rtimes(x)$	⊤	⊥	⊥	⊥	⊥
$P_a(x)$	⊥	⊥	⊤	⊤	⊥
$P_b(x)$	⊥	⊤	⊥	⊥	⊥
$\ltimes(x)$	⊥	⊥	⊥	⊥	⊤
$P'_a(x)$	⊥	⊥	⊥	⊤	⊥
$P'_b(x)$	⊥	⊤	⊤	⊥	⊥
output	⋊	b	b	a	⋉

Figure 3: Logical transducer in (1) over $\mathcal{M}(baa)$.

(1) *Logical transducer which expresses* $a \to b/b\_\_$

$$P'_a(x) = P_a(x) \wedge \neg P_b(p(x))$$
$$P'_b(x) = P_b(x) \vee \big(P_a(x) \wedge P_b(p(x))\big)$$

The equation for $P'_a(x)$ in (1) expresses the following: the index x will carry an 'a' in the output model iff it carries an 'a' in the input model, and the index preceding it does not carry a 'b' in the input model. Stated more simply, x will output as an 'a' iff it is an 'a' that is not preceded by a 'b' in the input. Similarly, $P'_b(x)$ says: x will carry a 'b' in the output iff it is either a 'b' in the input, or it is an 'a' that is preceded by a 'b'. The string transformation $baa \mapsto bba$ is presented in Figure 3. In the input model $\mathcal{M}(baa)$, the logical formula for $P'_a(2)$ evaluates to $\bot$ and the logical formula for $P'_b(2)$ evaluates to $\top$. These facts together capture the following relationship between the input and output models: the 'a' at index 2 of the input model becomes a 'b' in the output model.

More recent work within model-theoretic phonology uses the framework of Boolean Monadic Recursive Schemes (BMRS) to represent string functions. Logical transductions over monadic string models within the BMRS framework have been used to represent phonological maps (Chandlee and Jardine, 2021; Jardine and Oakden, 2023), model process interaction (Oakden, 2021), and provide logical characterizations of the expressivity of phonological maps (Bhaskar et al., 2020, 2023; Yolyan, 2025; Chandlee and Lindell, forth.). This framework expresses the predicates in the output signature using an `if...then...else` syntax; logical transductions within the BMRS

framework are referred to as *programs*. The logical transducer in (1) can equivalently be expressed as the BMRS program in (2).

(2) *Logical transducer with BMRS syntax*

$$P'_a(x) = \text{if } P_b(p(x)) \text{ then } \bot \text{ else } P_a(x)$$
$$P'_b(x) = \text{if } P_a(x) \text{ then } P_b(p(x)) \text{ else } P_b(x)$$

The equation for $P'_b(x)$ in (2) expresses the following: if x is an 'a' in the input, then it will output as 'b' iff it is preceded by a 'b'; otherwise, it will output as an 'b' iff it is a 'b' in the input. Chandlee and Jardine (2021) discuss how the `if...then...else` syntax of BMRS can be used to represent meaningful phonological generalizations such as licensing/blocking structures and elsewhere conditions. Section 4.1 of this paper discusses how this syntax is also valuable for expressing the generalizations involved in learning phonological maps. Thus, while these logical transductions can be expressed using propositional logic operators, the syntax of BMRS programs is useful for refining the learning problem.

2.3 Feature Models

The string models discussed so far have the requirement that the predicates $\{\rtimes, \ltimes, P_\sigma\}_{\sigma\in\Sigma}$ partition the domain. Phonological maps, however, target particular features or bundles of features (Jakobson et al., 1952; Clements and Hume, 1995). In order to model phonological words and maps, we use *unconventional* string models (Strother-Garcia et al., 2016), in which more than one predicate can hold at each index.[2] In the case where the predicates represent phonological features, we will refer to these unconventional string models as 'feature models'. In this case, the monadic predicates range over an alphabet of phonological features $\mathbb{F}$, rather than an alphabet of characters. For simplicity, we use the same notation for each feature $F \in \mathbb{F}$ as the associated monadic predicate. That is, $F(x) = \top$ means the sound at index x of the model has feature [+F] and $F(x) = \bot$ means it has feature [-F].[3] Moreover, because the requirement that only one predicate hold at each index is dropped for feature models, the predicates INITIAL/FINAL or MIN/MAX can be used in place of the boundary symbols $\rtimes/\ltimes$. For simplicity of demonstration, we consider here the limited collection of sounds /p,b,t,d,a,h/ over the four binary phonological features [sonorant], [coronal], [continuant], and [voice]. Each of these sounds corresponds with a unique combination of these four feature specifications, given in Figure 4. The sound /d/ for example, has the specification [−son, +cor, −cont, +voi]. Moreover, because postnasal voicing does not distinguish between nasal sounds, the feature [nasal] will be used to identify *any* nasal sound.

	p	b	t	d	a	h	N
[son]	−	−	−	−	+	−	+
[cor]	−	−	+	+	−	−	
[voi]	−	+	−	+	+	−	+
[cont]	−	−	−	−	+	+	−
[nas]	−	−	−	−	−	−	+

Figure 4: Limited inventory of sounds and features

A logical transducer over feature models specifies output predicates $F'(x)$ for every feature $F \in \mathbb{F}$. The logical transducer which expresses the postnasal voicing map in Zoque is given in (3). A sample transduction of /mpama/→[mbama] 'my clothing' is presented in Figure 5.

(3) *Logical transducer for Zoque postnasal voicing*

$$[voi]'(x) = [voi](x) \vee (\neg[cont](x) \wedge [nas](px))$$
$$F'(x) = F(x) \quad \text{for all other } F \in \mathbb{F}$$

input	m	p	a	m	a
	0	1	2	3	4
`Initial`(x)	$\top$	$\bot$	$\bot$	$\bot$	$\bot$
[son](x)	$\top$	$\bot$	$\top$	$\top$	$\top$
[cor](x)	$\bot$	$\bot$	$\bot$	$\bot$	$\bot$
[voi](x)	$\top$	$\bot$	$\top$	$\top$	$\top$
[cont](x)	$\bot$	$\bot$	$\top$	$\bot$	$\top$
[nas](x)	$\top$	$\bot$	$\bot$	$\top$	$\bot$
`Final`(x)	$\bot$	$\bot$	$\bot$	$\bot$	$\top$
[voi]$'(x)$	$\top$	$\top$	$\top$	$\top$	$\bot$
output	m	b	a	m	a

Figure 5: Logical transducer in (3) over $\mathcal{M}(mpama)$

In order to represent feature models more compactly, we use feature matrices to encode all the relevant information in a feature model. The input feature model for /mpama/ in Figure 5, for example, can be represented with the shorthand in (4).

[2] String models in which exactly one predicate holds at each index are referred to as 'conventional models' in previous research because that is the convention used in computer science literature (e.g. Buchi, 1960).

[3] This convention is not the only way to model phonological maps within this framework. We may instead allow $\mathbb{F}$ to contain both $[+F]$ and $[-F]$ as predicates. Moreover, Chandlee and Jardine (2021, pg.42) show that feature models can be adapted for non-binary feature systems as well. Further discussion of feature systems from a model-theoretic perspective can also be found in Nelson (2022).

(4) *Shorthand notation for* $\mathcal{M}(mpama)$

$$\begin{bmatrix}+\text{son}\\-\text{cor}\\+\text{voi}\\-\text{cont}\\+\text{nas}\end{bmatrix}\begin{bmatrix}-\text{son}\\-\text{cor}\\-\text{voi}\\-\text{cont}\\-\text{nas}\end{bmatrix}\begin{bmatrix}+\text{son}\\-\text{cor}\\+\text{voi}\\+\text{cont}\\-\text{nas}\end{bmatrix}\begin{bmatrix}+\text{son}\\-\text{cor}\\+\text{voi}\\-\text{cont}\\+\text{nas}\end{bmatrix}\begin{bmatrix}+\text{son}\\-\text{cor}\\+\text{voi}\\+\text{cont}\\-\text{nas}\end{bmatrix}$$

2.4 Subfactors

The substructures which are relevant to this paper are *subfactors* of monadic feature models. This section does not delve into formal definitions of subfactors in general; discussions and definitions of subfactors can be found in Rogers and Lambert (2019); Rawski (2021); Chandlee et al. (2019); Strother-Garcia et al. (2016). The relevant result is that subfactors of monadic *string* models corresponds with substring models. The substrings of $w \in \Sigma^*$ are the strings $v \in \Sigma^*$ for which there exist $x, y \in \Sigma^*$ such that $w = xvy$. For example, the substrings of 'bac' are $\{\lambda, b, a, c, ba, ac, bac\}$. The *subfactors* of $\mathcal{M}(w)$ are the models $\mathcal{M}(v)$ such that v is a substring of w.

The subfactors of *unconventional* string models, however, are more complex than substring models because any number of predicates can be true at each index. Figure 6 presents the subfactors of a feature model with a domain that has cardinality 1. The feature [coronal] is left out of this figure in order to keep the space manageable. An example of a subfactor of [-nas,-son,-cont,-voi] is a model with no specification for [voice]: [-nas,-son,-cont]. These structures no longer represents a particular sound, but instead collections of sounds. The subfactor [-nas,-son,-cont] corresponds with the set of sounds {/p/, /b/, /t/, /d/}. Although the space of subfactors for feature models is very large, they allow for a formal representation of relevant linguistic generalizations because each subfactor in Figure 6 represents a natural class.

For any model $\mathcal{M}$, the set of subfactors is denoted $Subfact(\mathcal{M})$. A subfactor is called a k-factor if the domain has cardinality k. The set of k-factors of $\mathcal{M}$ is denoted $Subfact_k(\mathcal{M})$. The subfactor relation between models is denoted $\sqsubseteq$.

2.5 Partially-Ordered Sets

A structure $(X, \leq)$ is a partially-ordered set (poset) iff $\leq$ is reflexive, antisymmetric, and transitive. Figure 6 is an example of a poset. Chandlee et al. (2019) show that this is true in general; for a model $\mathcal{M}$, the structure $(Subfact(M), \sqsubseteq)$ is a poset.

Two types of posets which are particularly relevant for the discussion of learning in this paper

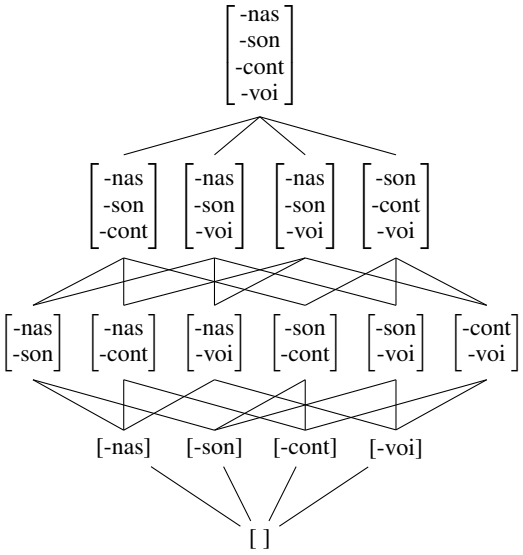


Figure 6: Poset of subfactors for feature models.

are filters and ideals, defined in Definitions 4 and 5, respectively. For every $x \in X$ of a poset, the subset of objects above x forms a filter, and the set of objects below x forms an ideal. The remainder of this paper shows how filters and ideals are used to generate and prune a hypothesis space.

Definition 4 (Filters). *For a poset $(X, \leq)$, a filter is a non-empty set $F \subseteq X$ such that:*
(a) for every $x \in F$, if $x \leq y$ then $y \in F$
(b) for every $x, y \in F$, there is some $z \in F$ such that $z \leq x$ and $z \leq y$
*The **principal filter** generated by $x \in X$ is the set*

$$\uparrow x := \{y \in X | x \leq y\}$$

Definition 5 (Ideals). *For a poset $(X, \leq)$, an ideal is a non-empty set $I \subseteq X$ such that:*
(a) for every $x \in I$, if $y \leq x$ then $y \in I$
(b) for every $x, y \in I$, there is some $z \in I$ such that $x \leq z$ and $y \leq z$
*The **principal ideal** generated by $x \in X$ is the set*

$$\downarrow x := \{y \in X | y \leq x\}$$

Posets have been used as a hypothesis space for various learning problems within phonology (Rawski, 2021; Chandlee et al., 2019; Tesar, 2013; Heinz et al., 2012; Heinz, 2010). A partially-ordered hypothesis space encodes entailment relations that are relevant for learning, which allow a large space to be pruned efficiently with a small number of data points. The next section presents two such examples to highlight this point.

3 Phonological Learning with Partially-Ordered Structures

Tesar (2013) uses a stress-length system to illustrate a learning algorithm for underlying forms with a partially-ordered hypothesis space. In this system every vowel in a word can be stressed or unstressed, and long or short. Consider the surface form [paká:], where the second vowel is both stressed and long. The space of possible underlying forms of [paká] is a poset under the *relative similarity* relation, defined as follows: $x \leq y$ iff y is more similar to [paká:] than x. This poset encodes entailment relations relevant to learning. If the map /paká/→[paká:] leads to inconsistency, then /paká/ is not a possible underlying form. This information entails that anything *less similar to* [paká:] than /paká/ will lead to inconsistency (in an output-driven map). Thus, being inconsistent is *downward entailing*, as illustrated in Figure 7 with the candidates marked ✗.

Although the poset contains 16 possible candidates for the underlying form of [paká:] (Figure 8), downward entailment of inconsistency allows the space to be pruned efficiently. If the learner determines that /paká/ cannot be the underlying form, the entire substructure highlighted in gray (i.e. the principal ideal ↓/paká/) in Figure 8 can be removed from the hypothesis space. As Tesar (2013) explains, the benefit of the structured hypothesis space is computational efficiency:

> [...]the possible optimality of a node entails upward, while the definite non-optimality of a node entails downward. This makes it possible to draw conclusions about whole sublattices of candidates based on the evaluation of only a single candidate. [...] This converts exponential search into linear search: the number of possible underlying forms is exponential in the number of features, but the number of forms to actually be tested is linear in the number of features. (Tesar, 2013, pg.287-289)

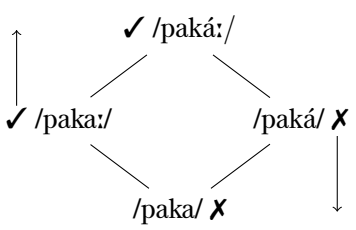


Figure 7: Entailments for learning underlying forms

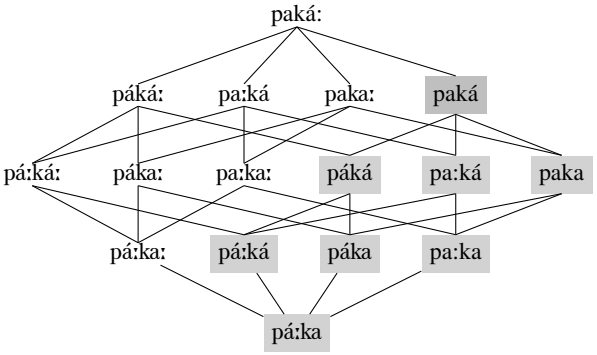


Figure 8: Removing ↓/paká/ from hypothesis space.

Similar observations are made by Chandlee et al. (2019) for the problem of learning phonotactic grammars. In this case, subfactors form a poset of potential banned structures in a grammar. Ungrammaticality is upward entailing; if a structure is ungrammatical, then all its superfactors must be ungrammatical. For example, if [+nas][-voi,-cont] is an ungramamtical surface form (as in the case of Zoque), then [+nas][-nas,-voi,-cont] must also be ungrammatical. On the other hand, grammaticality is downward entailing; if a structure is grammatical, then all its subfactors must also be grammatical. These entailments are illustrated in Figure 9, where ungrammatical structures are represented by ✗ and grammatical structures are represented by ✓. Similar to the previous example, these entailments are used to prune the hypothesis space. In this case, if a structure is ungrammatical, all the structures above it (i.e. the principal filter) will be removed from the hypothesis space. This process is discussed in the next section.

3.1 Bottom-Up Feature Inference Algorithm (BUFIA)

The learning algorithm introduced by Chandlee et al. (2019) is called the Bottom-Up Factor Inference Algorithm (BUFIA). The goal of BUFIA is

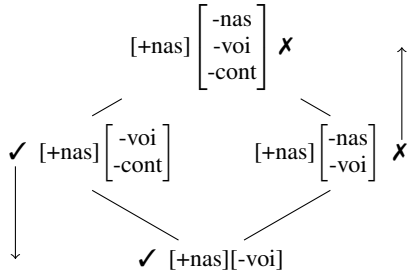


Figure 9: Entailments for learning grammars

to learn the *banned* k-factors of a language from a positive sample (i.e. collection of grammatical words from the language). BUFIA runs as follows. There is a finite sample S containing words from a language. The goal is to learn the *grammar* of the language, which is represented as a set of k-factors which are not permitted in the language. Because there are many ways to express the ungrammatical k-factors, a further goal is to find the most general description. For this reason, the algorithm searches through the hypothesis space *bottom-up*. If some k-factor x in the space is not found in any surface form in S, then it is added to the set of constraints. Because ungrammaticality is upward entailing, all of x's superfactors must also be ungrammatical. Thus, the entire filter $\uparrow x$ is removed from the hypothesis space since the grammaticality of these structures no longer needs to be considered. In this way, only the minimal (most general) ungrammatical k-factors are part of the resulting grammar.

BUFIA is illustrated here with the example of postnasal voicing in Zoque. For brevity, the features [sonorant] and [coronal] are left out of the discussion because they are ultimately not necessary to identify the minimal ungrammatical 2-factor in Zoque.[4] The surface forms from Section 1 provide evidence that all the subfactors of [+nas][-voi,-cont] are grammatical. These subfactors are presented in Figure 10 with N, V, and C for the features [nasal], [voi], and [cont], respectively. The form [tatah] 'father' provides evidence that [][-voi,-cont] is a grammatical structure. The form [nhayah] 'my husband' provides evidence that [+nas][-voi] is grammatical. The form [ndatah] 'my father' provides evidence that [+nas][-cont] is grammatical. Moreover, all the subfactors of these three 2-factors are grammatical. The 2-factor [+nas][-voi,-cont] is the *minimal* structure in the poset that is not found in any surface form of Zoque; all the structures below it are grammatical, while all the structures above it ungrammatical.

These data points are revisited in the next section

[4]The [-voi,-son] /h/ in [nhayah] 'my husband' does not have a voiced counterpart in Zoque, and therefore does not become voiced after a nasal (Wonderly, 1951). Thus, [nhayah] provides evidence that [+nas][-voi,-son] (and every subfactor of it) is grammatical. The feature [cont] is therefore necessary to distinguish the sounds which can be preceded by a nasal from the sounds which cannot. Moreover, since the ungrammaticality of [+nas][-voi,-cont] entails the ungrammaticality of both [+nas][-voi,-son,-cont] and [+nas][-voi,-cor,-cont], [son] and [cor] are not necessary to represent the ungrammatical structures in Zoque because a more general subfactor suffices.

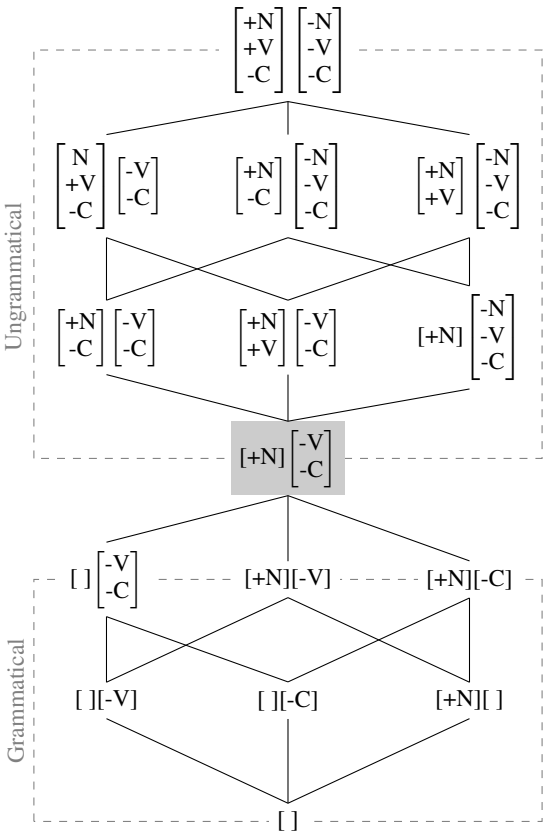


Figure 10: Minimal ungrammatical structure in Zoque.

to show how the same structures and principles can be used to learn the postnasal voicing map. Further discussion of BUFIA and abduction principles used in learning can be found in Rawski (2021). Recent applications of BUFIA can also be found in Li (2025) and Payne (2024).

As a final note, the learning problem for BUFIA (Def. 9 of Chandlee et al., 2019) is not stated in terms of finding the *correct* grammar in the limit, as in Gold (1967), but instead finding the most *general* grammar that is consistent with the sample of surface forms. The goal of learning phonological maps is similar: generate a logical transducer that expresses the most general phonological map that is consistent with the sample of input-output pairs. A formal statement of the learning problem in this paper is left for future work.

4 Learning Logical Transducers

Chandlee and Lindell (forth.) show that non-recursive quantifier-free logical transductions over monadic models are the logical characterization of finite-to-one ISL functions. A string function $f : \Sigma^* \to \Gamma^*$ is finite-to-one iff for every $w \in \Gamma^*$, the set $\{u \in \Sigma^* | f(u) = w\}$ is finite. In other

words, every surface form has finitely many underlying forms that map to it under the application of f. The output of the learning procedure introduced here is a quantifier-free BMRS program. This means we assume that we are always learning finite-to-one phonological maps. We moreover restrict the maps we consider to those that order-preserving and length-preserving. That is, epenthesis, deletion, and metathesis are not considered here. Future extensions to non-length-preserving maps are discussed in Section 5.

The examples from Tesar (2013) and Chandlee et al. (2019) illustrate how two very different learning goals can be pursued with partially-ordered hypothesis spaces, where entailment relations encoded in the structure make it possible for the learner to remove large substructures with a single data point. The remainder of this paper shows that the same observation holds of the problem of learning environments where features undergo change, and ultimately logical transducers which express a phonological map. The main idea expressed in this section is that learning a phonological map amounts to learning several phonotactic grammars simultaneously. In that sense, the learning procedure here is a way of *lifting* the ideas in Chandlee et al. (2019) to maps. This idea is formalized with the normal form for logical transducers proposed in Definition 6 and the main result in Theorem 7.

4.1 Normal Form for Logical Transductions

Consider a signature $\mathcal{S} = \{P_1, \dots P_n, s, p\}$, where each P_i is a unary predicate. A *term* over this signature is recursively defined as follows: a variable is a term; for every term T, $p(T)$ and $s(T)$ are terms. The *atoms* over this signature are defined as

$$atoms(\mathcal{S}) := \{P_i(T) \mid T \text{ is a term}\}$$

Definition 6. *Let $\{P'_\sigma\}_{\sigma\in\Sigma}$ be a collection of formulas defined over an input signature Σ. Each P'_σ is in **normal form** if it is expressed as*

$$P'_\sigma(x) = \textit{if } P_\sigma(x) \textit{ then } \neg\phi_\sigma(x) \textit{ else } \psi_\sigma(x)$$

where $\phi_\sigma(x)$ and $\psi_\sigma(x)$ are either $\top$, $\bot$, or disjunctive normal form expressions over $Atoms(\Sigma)$.[5]

Although normal form programs use the `if...then...else` syntax, the 'then' and 'else' parts of the equations are expressed with propositional operators.[6] This brings into question why we would bother using the `if...then...else` syntax at all. The reason for this representations is that the syntax compartmentalizes each equation in a program into two components: the 'then' expression and the 'else' expression. We consider first two examples which illustrate the information the 'then' and 'else' parts of these equations encode.

The normal form of the program in (2) is presented in (5). Recall that this program expresses the same function as the rewrite rule $a \to b / b\_\_$. The four pieces of information contained within this normal form program are presented in (6). The expression $\phi_a(x)$ is a logical description of the environment where an 'a' input becomes a non-'a' output. $\psi_b(x)$ is a logical description of the environment where a non-'b' input becomes a 'b' output. Similarly, $\psi_a(x) = \bot$ and $\phi_b(x) = \bot$ encode the facts that an non-'a' input never becomes an 'a', and a 'b' input never becomes a non-'b', respectively. The normal form in (5) rearranges the atoms of (2) into a syntactic form that explicitly references environments where change takes place.

(5) *Logical transducer from (2) in normal form*

$$\begin{aligned} P^*_a(x) &= \text{if } P_a(x) \text{ then } \neg\, P_b(px) \text{ else } \bot \\ P^*_b(x) &= \text{if } P_b(x) \text{ then } \neg\bot \\ &\quad \text{else } \big(P_a(x) \wedge P_b(px)\big) \end{aligned}$$

(6) *Information encoded in the normal form in (5)*

$$\begin{aligned} \phi_a(x) &= P_b(px) & \psi_a(x) &= \bot \\ \phi_b(x) &= \bot & \psi_b(x) &= P_a(x) \wedge P_b(px) \end{aligned}$$

A similar translation can be done for the postnasal voicing transducer in (3), presented in (7). The highlighted expression $[nas](px) \wedge \neg[cont](x)$ is a logical description of the environment in which a [-voi] sound in the input model becomes [+voi] in the output (i.e. $\psi_{[voi]}(x)$).

(7) *Logical transducer from (3) in normal form*

$$\begin{aligned} [voi]^*(x) &= \text{if } [voi](x) \text{ then } \neg\bot \\ &\quad \text{else } \big([nas](px) \wedge \neg[cont](x)\big) \\ F^*(x) &= \text{if } F(x) \text{ then } \neg\bot \text{ else } \bot \end{aligned}$$

The intuition for the normal form in Definition 6 is that every string function can be uniquely-identified with $\{\phi_\sigma, \psi_\sigma\}_{\sigma\in\Sigma}$, which encode information about the changes the function induces, and the environments those changes take place. With

[5] We assume that the input and output alphabets are the same. This means that we assume the underlying and surface forms have the same feature inventory. This assumption is not necessary, and is only used for simplicity.

[6] Every propositional operator can be expressed using `if...then...else`. For example, $p \vee q =$ if p then $\top$ else q.

respect to learning, the significance of the normal form is that it refines the learning problem in the following way: in order to learn a program $\{P'_\sigma\}_{\sigma\in\Sigma}$, it is sufficient to learn $\{\phi_\sigma, \psi_\sigma\}_{\sigma\in\Sigma}$. This fact is the immediate result of Theorem 7.

Theorem 7. *Every program $\{P'_\sigma\}_{\sigma\in\Sigma}$ over an input signature $\Sigma = \langle P_\sigma, s, p\rangle_{\sigma\in\Sigma}$ is logically equivalent to a program $\{P^*_\sigma\}_{\sigma\in\Sigma}$ where each P^*_σ is in normal form.*

The proof of Theorem 7 is presented in the appendix. The main idea is that for any output predicate P'_σ, there is a systematic way to construct an equivalent normal form predicate P^*_σ by generating a truth table of all the atoms in the definition of P'_σ, and determining ϕ_σ and ψ_σ from that truth table.

For phonological maps, learning a program over feature models amounts to learning $\{\phi_F, \psi_F\}_{F\in\mathbb{F}}$, where ϕ_F (ψ_F) expresses the environment where a [+F] ([-F]) sound in the underlying form becomes [-F] ([+F]) in the surface form. In this way, learning programs in this normal form amount to learning *phonological generalizations* regarding environments where features undergo change. The observation that connects this learning goal to the examples presented in Section 3 is that the space of possible expressions for ϕ_F and ψ_F can also be modeled as a poset, with relevant entailment relations that allow the space to be pruned efficiently.

Consider the case of postnasal voicing and the goal of learning the environment where a [-voice] input becomes [+voice] ($\psi_{[voi]}$). A similar diagram to Figures 7 and 9 is presented in Figure 11 for the goal of learning the environment where voicing takes place. Environments where voicing takes place are upward entailing, while environments where voicing does *not* take place are downward entailing. For example, if being a [-cont] that is immediately preceded by a [+nas] is sufficient to trigger voicing, then being a [-cont,-son] that is immediately preceded by a [+nas] is also sufficient. This is illustrated in Figure 11 with the logical expressions marked ✓. This means that in order to find the most general description of the environment that triggers voicing, we must traverse the space *bottom-up*. On the other hand, if an input-output pair indicates that some environment does *not* trigger voicing, then none of its subfactors are going to trigger voicing and therefore, its entire principal ideal can be removed from the hypothesis space, similar to the example of learning underlying forms in Figure 8.

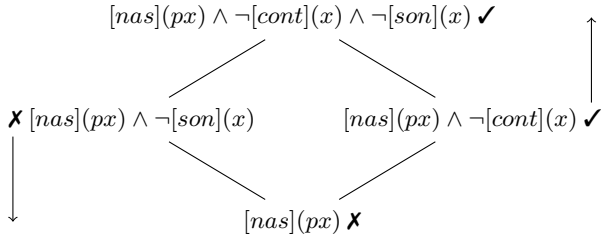


Figure 11: Entailments for learning environments

Each logical expression in Figure 11 corresponds with a description of a 2-*factor*. The expression $[nas](px) \wedge \neg[cont](x)$, for example, describes the 2-factor [+nas][-cont]. The subfactors which are used to learn grammars in BUFIA can therefore be adapted to learning environments. The distinction is that when we are learning *maps* rather than surface constraints, some index of the subfactor is the target of the map. Every k-factor we consider will therefore have an extra predicate `target` which identifies the unique index of the structure that is the target of the feature change in question. Figure 12 presents the 2-factor [+nas][-voi,-cont] with an additional predicate `target` which picks out the index that the voicing process targets. As a shorthand, we underline the subfactor that is the target of the map, as (8).

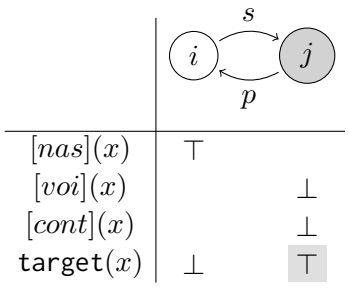


Figure 12: 2-factor targeted by postnasal voicing

(8) *Shorthand notation for 2-factor in Figure 12*

$$[+nas]\underline{\begin{bmatrix}-cont\\-son\end{bmatrix}}$$

Note that the subfactor partial order extends to a natural ordering on environments. For example, [+nas]<u>[-cont]</u> is a subfactor of [+nas]<u>[-cont,-son]</u>, but <u>[+nas]</u>[-cont] is not. Thus, the resulting structures with the `target` predicate form a partially ordered hypothesis space consisting of *environments*.

4.2 Variables and Initialization

In the case of learning grammars, there is a single hypothesis space consisting of all the possible subfactors. In the case of learning maps, there are several hypothesis spaces: one for each of the learning targets $\{\phi_F, \psi_F\}_{F\in\mathbb{F}}$. Thus, learning a program can be viewed as learning several grammars simultaneously. This section shows how principal ideals are used to construct each of the hypothesis spaces.

For every $X \in \{\phi_F, \psi_F\}_{F\in\mathbb{F}}$ we need two sets of k-factors: $V(X)$ and $\widehat{V}(X)$. $V(X)$ contains the relevant k-factors in which the targeted change takes place. $V(\phi_F)$, for example, contains all the k-factors in which an index x is such that $F(x) = \top$ in the input model but $F'(x) = \bot$ in the output model. $\widehat{V}(\phi_F)$ contains all the k-factors in which an index x is such that $F(x) = \top$ in the input model and $F'(x) = \top$ in the output model. In other words, $V(\phi_F)$ contains all the k-factors in which a [+F] sound becomes [-F], and $\widehat{V}(\phi_F)$ contains all the k-factors in which [+F] sounds do not undergo change. In this way, the collections V and $\widehat{V}$ encode positive and negative evidence for the environments where features undergo change. The hypothesis spaces are then defined in Definition 8.

Definition 8. *For every $X \in \{\phi_F, \psi_F\}_{F\in\mathbb{F}}$, the corresponding **hypothesis space** is the set*

$$\mathcal{H}(X) := \bigcup_{x\in V(X)} \downarrow x - \bigcup_{x\in\widehat{V}(X)} \downarrow x$$

subject to the following two restrictions:

(i) Every $\mathcal{M} \in \mathcal{H}(X)$ has exactly one index i in the domain of $\mathcal{M}$ such that $\texttt{target}(i) = \top$.

(ii) If X is ϕ_F for some $F \in \mathbb{F}$, then for every $\mathcal{M} \in \mathcal{H}(X)$, $F(i) = \top$ must hold for the unique i such that $\texttt{target}(i) = \top$. If X is ψ_F, then $F(i) = \bot$ must hold.

The posets defined in Definitions 8 takes all the k-factors in which the relevant change was observed in some pair of input-output models, and removes the subfactors in which the relevant change was *not* observed. The restrictions in (i) and (ii) then ensure that only the subfactors which represent relevant environments are included in the space. The restriction in (i) ensures that every structure in the hypothesis space represents an *environment*. The restriction in (ii) ensures that the hypothesis space includes only the *relevant* environments. If the goal is to learn the environment where a [-voi] sound becomes [+voi], then the hypothesis space should only include environments where the target of the change is [-voi]. This means that $\mathcal{H}(\psi_{[voi]})$ will only include k-factors where the unique index that satisfies the predicate `target` is such that $[voi](x) = \bot$.

The learning procedure starts with an initial hypothesis, and generates a new hypothesis with each input-output pair of models. Before any data have been observed, $V(X)$ and $\widehat{V}(X)$ are empty for each $X \in \{\phi_F, \psi_F\}$. This means that the initial hypothesis space for each X will also be empty by definition. Consequently the set of minimal elements, will be empty. In this case, the corresponding logical formula for X must be $\bot$. The initial variables, hypothesis, and corresponding BMRS program are summarized in (9). The consequence of V and $\widehat{V}$ being empty is that the initial (null) hypothesis corresponds with a BMRS program that represents the identity map.

(9) *Initial Variables and Hypothesis*

Variables; for all $X \in \{\phi_F, \psi_F\}_{F\in\mathbb{F}}$

$$\begin{aligned} V(X) &= \emptyset \\ \widehat{V}(X) &= \emptyset \\ min(\mathcal{H}(X)) &= \emptyset \end{aligned}$$

Individual learning goals

$$\{\phi_F(x) = \bot;\ \psi_F(x) = \bot\}_{F\in\mathbb{F}}$$

Initial Hypothesis as a BMRS program

$$\begin{aligned} \{F'(x) &= \text{if } F(x) \text{ then } \neg\bot \text{ else } \bot\}_{F\in\mathbb{F}} \\ &\equiv \{F'(x) = F(x)\}_{F\in\mathbb{F}} \end{aligned}$$

4.3 Updating the Hypothesis

When an input-output pair provides evidence for a feature change, the corresponding hypothesis space is updated. For example, if a pair of models $(\mathcal{M}, \mathcal{M}')$ is such that some index x is [-F] in the input model but [+F] in the output model, the set $V(\psi_F)$ will be updated with the k-factor(s) of $\mathcal{M}$ which contain x as the target. If, on the other hand, some index x is [-F] in the input and remains [-F] in the output, the set $\widehat{V}(\psi_F)$ will be updated with the relevant k-factors. Each time V or $\widehat{V}$ is updated, the set of minimal elements must also be updated. Thus, there are two possible types of updates.

Consider first the case in which a k-factor x is added to $V(X)$ for some X. In this case, the hypothesis space $\mathcal{H}(X)$ grows, and potentially the set of minimal elements grow. This means that when a new k-factor x is added to $V(X)$, the update function maintain all previous minimal elements and only determines whether to add new ones. The pro-

cedure for determining which elements to add to $M(X)$ is as follows: we traverse the structure $\downarrow x$ bottom-up in order to find the minimal subfactors of x which are neither subfactors of any current minimal elements, nor subfactors of any k-factor in $\widehat{V}(X)$. Because both k and the number of features we consider are fixed, the structure $\downarrow x$ is always of fixed size. Moreover, we only need to know the set of factors in $M(X)$ and $\widehat{V}(X)$ in order to determine the new set of minimal elements.

In the case where a new k-factor x is added to $\widehat{V}(X)$, the update function must remove any element $m \in M(X)$ such that $m \sqsubseteq x$, and potentially add new minimal elements. In this case, there are two possibilities: if $x \in V(X)$, then m is removed from the set of minimal elements; otherwise m is replaced by new minimal elements that are immediately above m in the poset. In this case, we only need information about $V(X)$ and $M(X)$ in order to update the set of minimal elements.

Both types of updates therefore amount to traversing a restricted space in order to find new minimal elements. Although the hypothesis space can get very large, for each $X \in \{\phi_F, \psi_F\}$, only the sets $M(X)$, $V(X)$, and $\widehat{V}(X)$ need to be maintained in memory in order to update the hypothesis. The next section illustrates the update process with the example of postnasal voicing.

4.4 Case Study: Postnasal Voicing in Zoque

Consider first the input-output pair of models in Figure 5 representing /mpama/ → [mbama]. The input-output models for this transformation were presented in Figure 5. The sound at index 1 is [-voi] in the input model and [+voi] in the output model. This means that the set $V(\psi_{[voi]})$ must be updated with the 2-factors that contain the sound at index 1 as a target. There are two possible 2-factors in the input model which contain the sound at index 1. These are presented in Figure 13. The updated hypothesis space is presented in Figure 14. The corresponding updates are summarized in (10).

	m	p
[nas](x)	⊤	⊥
[voi](x)	⊤	⊥
[cont](x)	⊥	⊥
target(x)	⊥	⊤

	p	a
[nas](x)	⊥	⊥
[voi](x)	⊥	⊤
[cont](x)	⊥	⊤
target(x)	⊤	⊥

Figure 13: 2-factors added to $V(\psi_{[voi]})$

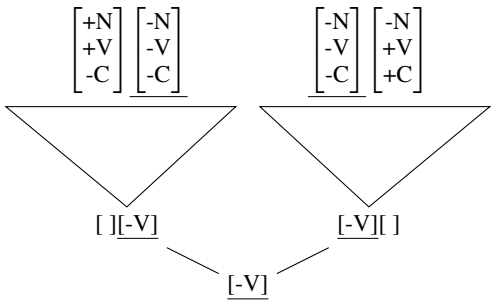


Figure 14: $\mathcal{H}(\psi_{[voi]})$ given (/mpama/, [mbama]).

Since being [-voi] is presumed to be true when learning $\psi_{[voi]}$ (by restriction (ii) of Definition 8), the minimal element is [-V] rather than $\emptyset$. This element corresponds with $\psi_{[voi]}(x) = \top$. We could also set $\psi_{[voi]} = \neg[voi](x)$; this would give a logically equivalent program and corresponding map, but with unnecessary redundancy. The new hypothesis therefore says that a [-voi] sound *always* becomes [+voi]. This hypothesis corresponds with the *most general* hypothesis that accounts for the input-output sample observed.

(10) *Hypothesis update given /mpama/→[mbama]*

$$min(\mathcal{H}(\psi_{[voi]})) = \{[-V]\}$$
$$\psi_{[voi]}(x) = \top$$

as a BMRS program

$[voi]^*(x) =$ if $[voi](x)$ then $\neg\bot$ else $\top$

as a phonological map

$[-voi] \rightarrow [+voi]$

We consider next the case where the underlying and surface forms are the same. Consider the surface form [tatah] 'father'. The hypothesis in (10) predicts that we should not see this surface form; we would instead expect to see [dadah]. This surface form therefore provides evidence that the the previous hypothesis is incorrect. In this case, the pair of 2-factors in Figure 15 are added to $\widehat{V}(\psi_{[voi]})$. The updated hypothesis space is presented in Figure 16. The corresponding updates are summarized in (11)

	a	t
[nas](x)	⊥	⊥
[voi](x)	⊤	⊥
[cont](x)	⊤	⊥
target(x)	⊥	⊤

	t	a
[nas](x)	⊥	⊥
[voi](x)	⊥	⊤
[cont](x)	⊥	⊤
target(x)	⊤	⊥

Figure 15: 2-factors added to $\widehat{V}(\psi_{[voi]})$

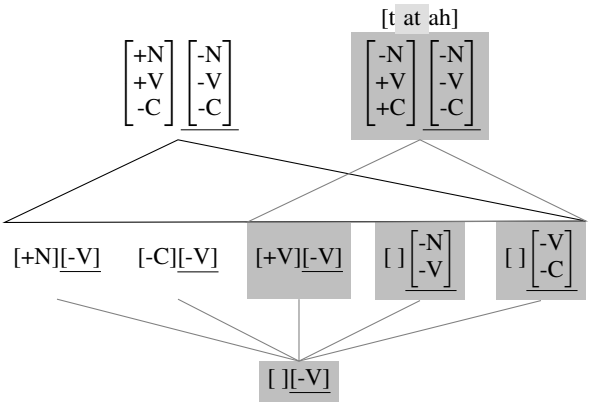


Figure 16: Updated $\mathcal{H}(\psi_{[voi]})$ given (/tatah/, [tatah]).

The hypothesis space has two minimal elements: [+N][-V] and $[-C][-V]$. The resulting hypothesis is therefore a *disjunction*. The new hypothesis says that a [-voi] sound becomes [+voi] if it is either preceded by a [+nas] *or* by a [-cont] sound. The remaining updates with /nhayah/→[nhayah] and /kuʔtpa/→[kuʔtpa] are presented in Figure 17.

(11) *Hypothesis update given /tatah/→[tatah]*

$$min(\mathcal{H}(\psi_{[voi]})) = \{[+N][-V], [-C][-V]\}$$
$$\psi_{[voi]}(x) = [nas](px) \vee \neg[cont](px)$$

as a BMRS program

$$[voi]^*(x) = \text{if } [voi](x) \text{ then } \neg\bot \text{ else } [nas](px) \vee \neg[cont](px)$$

as a phonological map

$$[-voi] \rightarrow [+voi] / \left\{ \begin{matrix} [+nas] \\ [-cont] \end{matrix} \right\} —$$

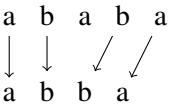
The remaining subfactors in the hypothesis space in Figure 17 are exactly the ungrammatical 2-factors as Figure 10. While [+N][-V,-C] is the most general description of ungrammatical 2-factors in Zoque, [+N][-V,-C] is the most general description of the environment where a sound becomes voiced. Since ungrammaticality is upward entailing (Figure 9), it makes sense that being an environment where voicing takes place is also upward entailing (Figure 11); a [-V,-C] segment becomes voiced when it is preceded by a nasal as a means of *repairing* the fact that [+N][-V,-C] is an ungrammatical surface form. The learning approach presented here therefore makes the relationship between surface constraints and phonological maps concrete.

5 Discussion and Conclusion

This paper starts with length-preserving and order-preserving ISL functions in order to show how partially-ordered hypothesis spaces over model-theoretic representations can be used to learn phonological maps as logical transducers. Chandlee and Jardine (2021, pg.3) state the appeal of the BMRS formalism for phonology as follows: "this formalism has a well-understood complexity bound that corresponds to previous results in the study of computational phonology, but it also provides a way to implement phonological substance". The approach presented here is extends this advantage of BMRS to learning; because BMRS can be used to represent phonological generalizations, we can adapt previous work on model-theoretic learning to show that BMRS can also be used *learn* phonological generalization. This latter point is ultimately the broader contribution of this work.

This paper assumed that there is a trivial one-to-one correspondence between input and output elements, where the segment at index x in the output model is the surface form of the segment at index x in the input model. This assumptions rules out deletion, epenthesis, and metathesis maps which are also ISL (Chandlee, 2014). An interesting extension of this work would be to develop a learning procedure for input-output index correspondences. Consider for example a simple deletion map $a \rightarrow \lambda / b\_b$, where an 'a' is deleted whenever it is between two 'b's. (12) presents the input-output correspondences for the string transformation $ababa \mapsto abba$.

(12) *Deletion input-output correspondences*



Recent work from Li (2025) uses BUFIA to learn tonotactic patterns, where the structures in the hypothesis space encode autosegmental representations. An interesting question to pursue further is whether a similar hypothesis space can be used to represent input-output correspondences. Doing so would amount to learning the environments where features undergo change and the input-output correspondence between segments in the underlying and surface forms simultaneously.

A further extension of this work is to extend hypothesis space to contain *pairs* of subfactors, corresponding with input and output string models. In this case, the corresponding BMRS programs would be recursive, allowing for representation of output strictly local maps (Chandlee et al., 2015).

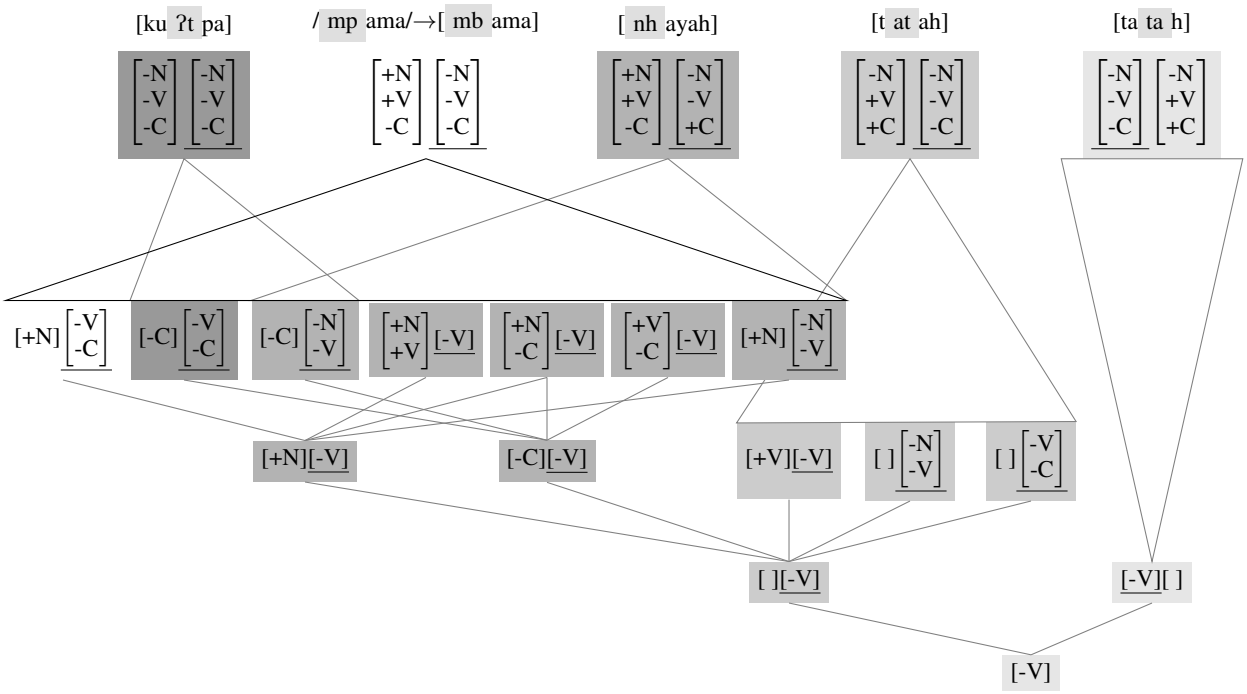


Figure 17: Updated $\mathcal{H}(\psi_{[voi]})$ given /nhayah/→[nhayah] and /kuʔtpa/→[kuʔtpa].

Theorem 7 stated that every non-recursive BMRS programs can be expressed as an equivalent normal form program. However, it seems intuitive that this result should extend to programs in general because *every* map can be uniquely identified in terms of environments where features undergo change; this is afterall how SPE-style rewrite rules express a function. However, such a result requires reasoning about recursion in BMRS, which uses a fixed point semantics (Bhaskar et al., 2020), and is outside the scope of this paper.

Ultimately, the purpose of this paper was to present a starting point for a model-theoretic approach to learning phonological maps, and the merits of such an approach. The ideas and procedure presented here can be extended to different representations and classes of functions. In order to extend this work to more complex representations, the next step for this research is an implementation of the learning procedure presented here. Doing so would also make it possible to obtain *empirical* results.

References

Siddharth Bhaskar, Jane Chandlee, and Adam Jardine. 2023. Rational functions via recursive schemes. *Preprint*, arXiv:2302.03074.

Siddharth Bhaskar, Jane Chandlee, Adam Jardine, and Christopher Oakden. 2020. Boolean monadic recursive schemes as a logical characterization of the subsequential functions. In *Language and Automata Theory and Applications (LATA)*, Lecture Notes in Computer Science, pages 157–169. Springer.

Richard Buchi. 1960. Weak second-order arithmetic and finite automata. *Mathematical Logic Quarterly*, 6:66–92.

Jane Chandlee. 2014. *Strictly Local Phonological Processes*. Ph.D. thesis, University of Delaware.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2014. Learning strictly local subsequential functions. *Transactions of the Association for Computational Linguistics*, 2:491–504.

Jane Chandlee, Rémi Eyraud, and Jeffrey Heinz. 2015. Output strictly local functions. In *Proceedings of the 14th Meeting on the Mathematics of Language*, page 112–125. Association for Computational Linguistics.

Jane Chandlee, Remi Eyraud, Jeffrey Heinz, Adam Jardine, and Jonathan Rawski. 2019. Learning with partially ordered representations. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 91–101, Toronto, Canada. Association for Computational Linguistics.

Jane Chandlee and Jeffrey Heinz. 2018. Strict locality and phonological maps. *Linguistic Inquiry*, 49(1):23–60.

Jane Chandlee and Adam Jardine. 2019a. Autosegmental input strictly local functions. In *Transactions of the Association for Computational Linguistics*, volume 7.

Jane Chandlee and Adam Jardine. 2019b. Quantifier-free least fixed point functions for phonology. In *Proceedings of the 16th Meeting on the Mathematics of Language (MOL)*, pages 50–62. Association for Computational Linguistics.

Jane Chandlee and Adam Jardine. 2021. Computational universals in linguistic theory: Using recursive programs for phonological analysis. *Language*, 97.

Jane Chandlee and Steven Lindell. forth. Logical perspectives on strictly local transformations. In *Doing Computational Phonology*. Oxford University Press.

Nick Clements and Elizabeth Hume. 1995. The internal organisation of speech sounds. In John Goldsmith, editor, *The Handbook of Phonological Theory*, chapter 7, pages 245–307. Blackwell.

Bruno Courcelle. 1994. Monadic second-order definable graph transductions: a survey. *Theoretical Computer Science*, 126(1):53–75.

Bruno Courcelle and Joost Engelfriet. 2012. Monadic second-order transductions. In *Graph Structure and Monadic Second-Order Logic: A Language-Theoretic Approach*, chapter 7, pages 505–577. Cambridge University Press.

Hossep Dolatian. 2020. *Computational locality of cyclic phonology in Armenian*. Ph.D. thesis, Stony Brook University.

Joost Engelfriet and Hendrik Jan Hoogeboom. 2001. Mso definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic (TOCL)*, 2(2):216–254.

Mark Gold. 1967. Language identification in the limit. *Information and Control*, 10:447–474.

Jeffrey Heinz. 2010. String extension learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics.

Jeffrey Heinz, Anna Kasprzik, and Timo Kötzing. 2012. Learning in the limit with lattice-structured hypothesis spaces. *Theoretical Computer Science*, 457.

Roman Jakobson, Gunnar Fant, and Morris Halle. 1952. *Preliminaries to Speech Analysis*. MIT Press.

Adam Jardine. 2017. The local nature of tone-association patterns. *Phonology*, 34:385–405.

Adam Jardine and Chris Oakden. 2023. Computing process-specific constraints. *Linguistic Inquiry*.

Dakotah Lambert and James Rogers. 2020. Tier-based strictly local stringsets: Perspectives from model and automata theory. In *Proceedings of the Society for Computation in Linguistics 2020*, pages 159–166. Association for Computational Linguistics.

Han Li. 2025. Learning tonotactic patterns over autosegmental. *Proceedings of the 2023 and 2024 Annual Meetings on Phonology*.

Leonid Libkin. 2004. *Elements of Finite Model Theory*.

David Marker. 2002. *Model Theory: An Introduction*.

Scott Nelson. 2022. A model theoretic perspective on phonological feature systems. In *Proceedings of the Society for Computation in Linguistics*, volume 5.

Chris Oakden. 2021. *Modeling phonological interactions using recursive schemes*. Ph.D. thesis, Rutgers University.

Sarah Payne. 2024. A generalized algorithm for learning positive and negative grammars with unconventional string models. In *Proceedings of the Society for Computation in Linguistics (SCiL)*, pages 75–85.

Jonathan Rawski. 2021. *Structure and Learning in Natural Language*. Ph.D. thesis, Stony Brook University.

James Rogers. 2003. Syntactic structures as multi-dimensional trees. *Research on Language and Computation*, 1:265–305.

James Rogers and Dakotah Lambert. 2019. Some classes of sets of structures definable without quantifiers. In *Proceedings of the 16th Meeting on the Mathematics of Language*, pages 63–77. Association for Computational Linguistics.

James Rogers and Geoffrey Pullum. 2011. Aural pattern recognition experiments and the subregular hierarchy. *Journal of Logic, Language and Information*, 20:329–342.

Kristina Strother-Garcia. 2018. Imdlawn Tashlhiyt Berber syllabification is quantifier-free. In *Proceedings of the Society for Computation in Linguistics*.

Kristina Strother-Garcia. 2019. *Using model theory in phonology: a novel characterization of syllable structure and syllabification*. Ph.D. thesis, University of Delaware.

Kristina Strother-Garcia, Jeffrey Heinz, and Hyun Jin Hwangbo. 2016. Using model theory for grammatical inference: a case study from phonology. In *Proceedings of The 13th International Conference on Grammatical Inference*, pages 66–78. PMLR.

Bruce Tesar. 2013. *Output-Driven Phonology: Theory and Learning*. Cambridge University Press.

Mai Vu, Ashkan Zehfroosh, Kristina Strother-Garcia, Michael Sebok, Jeffrey Heinz, and Herbert G. Tanner. 2018. Statistical relational learning with unconventional string models. *Frontier in Robotics and AI*.

William L. Wonderly. 1951. Zoque II: Phonemes and Morphophonemics. *International Journal of American Linguistics*, 17:105–123.

Tatevik Yolyan. 2025. A logical characterization of weak determinism as simultaneous application. *Journal of Logic, Language and Information*, 34(1):89–153.

A Appendix

Proof of Theorem 7. Let $\{P'_\sigma\}_{\sigma\in\Sigma}$ be a non-recursive BMRS program over the input signature $\Sigma = \{P_\sigma, p, s\}_{\sigma\in\Sigma}$. Fix $\sigma \in \Sigma$, and let $Atoms(P'_\sigma(x))$ be the set of Boolean expressions in $Atoms(\Sigma)$ which appear in the equation $P'_\sigma(x)$. We can then construct a truth table with columns $\mathbb{C} = atoms(P'_\sigma(x)) \cup \{P'_\sigma(x)\}$. For every $\alpha \in \mathbb{C}$ and row r, we say $r \models \alpha$ if and only if α evaluates to $\top$ in row r of the table. For every row r, we define a conjunction consisting of all the information contained in row r as follows

$$\Psi_r(x) := \bigwedge_{\alpha\in Atoms(P'_\sigma(x))} \begin{cases} \alpha & \text{if } r \models \alpha \\ \neg\alpha & \text{otherwise} \end{cases}$$

We show that there is a normal form expression P^*_σ such that P'_σ is logically equivalent to P^*_σ.
Case 1. Consider first the case where $P_\sigma(x) \notin \mathbb{C}$. In other words, the equation $P'_\sigma(x)$ is not defined in terms of the input predicate $P_\sigma(x)$. Set $\psi_\sigma(x)$ and $\phi_\sigma(x)$ as follows.

$$\phi_\sigma(x) := \bigvee_{\{r : r \not\models P'_\sigma(x)\}} \Psi_r(x)$$

$$\psi_\sigma(x) := \bigvee_{\{r : r \models P'_\sigma(x)\}} \Psi_r(x)$$

Let $P^*_\sigma(x) =$ if $P_\sigma(x)$ then $\neg\phi_\sigma(x)$ else $\psi_\sigma(x)$. By construction, $\phi_\sigma(x) = \neg\psi_\sigma(x)$, and therefore $P^*_\sigma(x) \equiv \psi_\sigma(x)$. Because $\psi_\sigma(x)$ exhaustively contains all the possible situations in which $P'_\sigma(x)$ evaluates to $\top$, $\psi_\sigma(x) \equiv P'_\sigma(x)$. Thus, P^*_σ is equivalent to P'_σ.
Case 2. $P_\sigma(x) \in atoms(P'_\sigma(x))$. Define $\phi_\sigma(x)$ and $\psi_\sigma(x)$ as follows.

$$\phi_\sigma(x) := \bigvee_{\{r : r \models P_\sigma(x),\, r \not\models P'_\sigma(x)\}} \Psi_r(x) \qquad (1)$$

$$\psi_\sigma(x) := \bigvee_{\{r : r \not\models P_\sigma(x),\, r \models P'_\sigma(x)\}} \Psi_r(x) \qquad (2)$$

Set $P^*(x) =$ if $P_\sigma(x)$ then $\neg\phi_\sigma(x)$ else $\psi_\sigma(x)$. We show that $P^*_\sigma(x)$ evaluates to $\top$ iff $P'_\sigma(x)$ evaluates to $\top$. Consider first the case where $P_\sigma(x)$ evaluates to $\top$. Then $P^*_\sigma(x)$ evaluates to $\top$ iff $\phi_\sigma(x)$ evaluates to $\bot$. By the definition in (1), this means Ψ_r evaluates to $\bot$ for every row r such that $r \not\models P'_\sigma(x)$. In other words, none of the conditions under which $P'_\sigma(x)$ can evaluate to $\bot$ are satisfied. Thus, $\phi_\sigma(x)$ evaluates to $\bot$ iff $P'_\sigma(x)$ evaluates to $\top$. Similarly in the case where $P_\sigma(x)$ evaluates to $\bot$, $P^*_\sigma(x)$ evaluates to $\top$ iff $\psi_\sigma(x)$ evaluates to $\top$. By the definition in (2), $\psi_\sigma(x)$ evaluates to $\top$ iff $\Psi_r(x)$ evaluates to $\top$ for *some* row r. In other words, one of the conditions under which $P'_\sigma(x)$ can evaluate to $\top$ is satisfied. Thus, $\psi_\sigma(x)$ evaluates to $\top$ iff $P'_\sigma(x)$ evaluates to $\top$. □